This document details deviations to the actual performance of the EDK product from that shown in the documentation supplied on this CD. These deviations are due to the behaviour of the Hardware Break Controller in H8/38124 devices, and the fact that HMON does not support interrupts with higher priorities than the serial port.

Hardware Break Controller

The hardware break controller for the H8/38124 devices only functions in BOOT mode. It is therefore necessary to return the device to BOOT mode every time the board is reset using the RES switch when debugging. After a successful BOOT mode download, FDT can be placed in User mode; however, when prompted to reset the board it must be reset into BOOT mode. FDT User mode flash programming can then be performed until the power is cycled on the board.

The hardware break also occurs one instruction after the break point. Therefore if a breakpoint is set to the beginning of main() as described in section 5.5 of the Tutorial Manual and Reset Go is pressed then the main function will be opened and the cursor will be set to point to the start of the main function.

Interrupt Priority

Because HMON does not work reliably when interrupts of a higher priority than the serial port are used, do not use interrupts other that the ADC interrupt.

Tutorial Software

Section 5.5.2

Because of the above limitation only the ADC interrupt is used in the tutorial software provided with the 3DK38124. Therefore the Timer and A/D test behaves in a different way in the tutorial software. Section 5.5.2 of the Tutorial Manual can be largely ignored.

Instead the function Timer_AD sets up ADC interrupts. As these occur periodically, they are used to decrement a variable, adccount. When this reaches zero, the value read from the ADC is used as the new adccount value.
In this way changing the potentiometer changes the frequency of the LEDs.

The last two paragraphs of 5.5.2 still apply.

Section 5.5.6

This function is replaced by SubCLK_Test(), which polls TimerA, the RTC timer, rather than use interrupts.
It counts on the LEDs in 1s intervals.

```
void SubCLK_Test(void)
{
      count = 0x07;
      #ifdef Release
   PutStr("\n\nSub-clock test in progress...\n");
      #endif

      // Count on the LEDs at 1 second intervals using the Timer
      while (count != 0)
      {
            PollTimer();
            LED_PORT_DR = ~count;
      }

      #ifdef Release
   PutStr("Sub-clock test complete.\n");
      #endif

}
```

This uses the function PollTimer():

```
void PollTimer(void)
{

	/* Poll the interrupt request bit*/
	while (	IRR1.BIT.IRRTA == CLEAR)
	{
		nop();
	}
	count--;
	IRR1.BIT.IRRTA = CLEAR;
}
```

This function waits in a loop until the interrupt request flag becomes true, decrements the counter, and clears the flag. Note that interrupts are nor enabled.